Enumeration rules

This document is part of the online appendix to the paper titled "Optimizing Navigational Graph Queries" and details the *enumeration rules* that were not presented in the paper. To this end, a few helpful definitions are given in section 1, the additional enumeration rules are presented in section 2 - 5 and the topic of projection (push-down) is discussed in section 6.

1 Definitions

Definition 1 *Let $Q(\overline{x}) \leftarrow R_1(\overline{y_1}), ..., R_n(\overline{y_n})$ be a query. The set of query variables Y_Q for Q is defined as:*

$$Y_Q = \bigcup_{1 \leq i \leq n} \overline{y_i}$$

Definition 2 *Let y_1, y_2 be query variables and $\theta \in \{=, \neq, <, >, \leq, \geq\}$ a comparator. A literal predicate $y_1 \, \theta \, y_2$ is a predicate that evaluates to true or false for any pair of bindings of y_1 and y_2.*

Definition 3 *Let $Q(\overline{x}) \leftarrow R_1(\overline{y_1}), ..., R_n(\overline{y_n}), y_{1,1} \, \theta_1 \, y_{1,2}, ..., y_{m,1} \, \theta_m y_{m,2}$ be a query. The set of literal predicates T_Q for Q is defined as:*

$$T_Q = \bigcup_{1 \leq j \leq m} \{y_{j,1} \, \theta_j \, y_{j,2}\}$$

2 Edges rule

This rule applies to abstractions of the form $\square(Q(\overline{x}) \leftarrow E(s, e, t))$. That is, it solves the recursive sub-problem which asks for all triples (s, e, t) that represent edges in the data graph. With $i \in \mathbb{N}$ as a fresh identifier to distinguish multiple instances of E, this rule outputs exactly one plan $\mathcal{P}$. If $\overline{x} = \{s, e, t\}$ then $\mathcal{P} = (\{E(i)\}, E(i))$. However, when $\overline{x} \subset \{s, e, t\}$ a projection operator is necessary and $\mathcal{P} = \{\Pi(\overline{x}, E(i)), E(i)\}, \Pi(\overline{x}, E(i)))$.

3 Properties rule

This rule applies to abstractions of the form $\square(Q(\overline{x}) \leftarrow P(o, k, v)$. That is, it solves the recursive sub-problem which asks for all triples (o, k, v) that represent an object's property values in the data graph. With $i \in \mathbb{N}$ as a fresh identifier to distinguish multiple instances of P, this rule outputs exactly one plan $\mathcal{P}$. If $\overline{x} = \{o, k, v\}$ then $\mathcal{P} = (\{P(i)\}, P(i))$. However, when $\overline{x} \subset \{o, k, v\}$ a projection operator is necessary and $\mathcal{P} = \{\Pi(\overline{x}, P(i)), P(i)\}, \Pi(\overline{x}, P(i)))$.

4 Union rule

This rule applies to abstractions of the form:

$$\begin{aligned} \square(Q(\overline{x}) &\leftarrow B_1 \\ &... \\ Q(\overline{x}) &\leftarrow B_n) \end{aligned}$$

where B_1 through B_n are non-recursive. That is, it solves the recursive sub-problem which asks for the union of the evaluations of B_1 through B_n projected to $\overline{x}$. The abstraction is split into n sub-problems of the form $Q_i(\overline{x}) \leftarrow B_i$ for $1 \leq i \leq n$ and a union operator is constructed as the parent of each of these abstractions. Hence, the rule outputs exactly one plan of the form:

$$\begin{tikzcd} & \cup \arrow[dl] \arrow[dr] & \\ \Box(Q_1(\overline{x})) \leftarrow B_1) & \cdots & \Box(Q_n(\overline{x})) \leftarrow B_n) \end{tikzcd}$$

Figure 1: Union plan

5 Selection rule

This rule applies to abstractions of the form $\Box(Q(\overline{x}) \leftarrow R_1(\overline{y_1}), ..., R_n(\overline{y_n}), y_{1,1}\,\theta_1\,y_{1,2}, ..., y_{m,1}\,\theta_m\,y_{m,2})$. That is, it solves the recursive sub-problem which asks for those tuples that satisfy a set of topological constraints captured by $R_1, ..., R_n$ and a set of literal constraints captured by $y_{1,1}\,\theta_1\,y_{1,2}, ..., y_{m,1}\,\theta_m\,y_{m,2}$.

Let T' be a set of literal predicates defined as follows:

$$T' = \{y_1\,\theta\,y_2 \,|\, y_1\,\theta\,y_2 \in T_Q \wedge \exists_{1 \leq i,j \leq n} \,|\, i \neq j \wedge y_1 \in \overline{y_i} \wedge y_2 \in \overline{y_j}\}$$

That is, T' is the subset of T_Q consisting only of all those literal predicates that reference variables y_1, y_2 that come from different $\overline{y_i}, \overline{y_j}$. These are the literal predicates that cannot be *pushed down* (e.g., below a join). This rule applies *only* to queries Q for which $T' \neq \emptyset$. The set T' is constructed and a selection operator is instantiated over it. The predicates in T' are removed from the input abstraction and the resulting abstraction is added as a child of the selection operator. Hence, the result outputs exactly one plan of the form:

$$\begin{tikzcd} \sigma\big(T'\big) \arrow[d] \\ \Box\big(Q(\overline{x}) \leftarrow R_1(\overline{y_1}), ..., R_n(\overline{y_n}), T_Q - T'\big) \end{tikzcd}$$

Figure 2: Selection plan

Additional projection may be required when the set of variables referenced in T' contains variables that are not in $\overline{x}$.

6 Projection

There is no enumeration rule that deals exclusively with projection. Instead, all rules are defined in such as way as to facilitate maximum *projection push-down*. That is, the join, seeding and selection rules all ensure that the output schemas $\overline{x}$ of the abstractions they instantiate are minimized (i.e., contain no variables that are not required higher up in the query plan). Projection operators are only instantiated once further push-down is no longer possible, for example as parents of leaf operators $E(i)$ or $P(i)$ or as parents of join operators.